CA213 Lab Test 2

This test is conducted under normal examination regulations. You may not access the web or any information on your computer other than in the directory in which you work. You may not communicate with other candidates. Mobile phones must be switched off. You may use a blank notebook in which to write rough work.

1. Credit will be based solely on the contents of the file `StringSetTest.java` which you should leave in your directory at the end of the test.
2. `StringSetTest.java` will contain classes `StringSet` and `StringSetTest`, and no others. Both classes must be in the same file.
3. Any code not in `StringSetTest.java` – with precisely that spelling and capitalisation – will not be marked.

The following ADT encapsulates a set of strings:

Class name: `StringSet`
Data: Set of (non-`null`) strings
Constructors: Default no-args constructor creating an empty set
Methods:

`int size();`
Number of elements in set

`void add(String s);`
Add `s` to set (if not already present). It is guaranteed that `s` is not `null`.

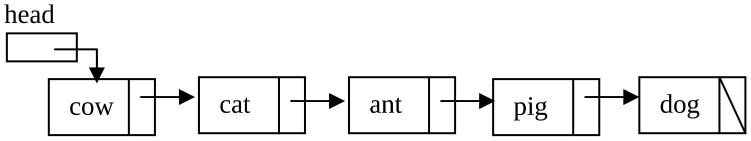
`void remove(String s);`
Remove `s` from set (if present). It is guaranteed that `s` is not `null`.

`boolean contains(String s);`
`s` in set? It is guaranteed that `s` is not `null`.

`void put();`
Print the elements of the set, with a space between each element

This is similar to sets presented in lectures, but is simpler in that you know that the elements of the set are of type `String` rather than a generic type `T`, and `add` and `remove` do not return a boolean indicating any change in the set.

You are given file `StringSetTest.java` containing a partial implementation of `StringSet` (reproduced below) based on singly linked lists (e.g. the implementation of a set with five elements might be represented pictorially thus):



Your task is to complete the implementation of `StringSet`.

```
class StringSet  {

  private static class Node {
    private String item;           // data
    private Node next = null;      // successor node

    Node(String item0, Node next0) {
      item = item0; next = next0;
    }
  }

  private Node head = null; // first node (null if empty)
  private int numItems = 0; // number of elements in set

  int size() {
    // code this
  }

  void add(String s) {
    // code this
  }

  void put() {
    // code this
  }

  boolean contains(String s) {
    // code this
  }

  void remove(String s) {
    // code this
  }
}
```

Note that you are asked only to complete the bodies of some methods. Do not change `StringSet` in any other way, neither changing it, adding to it, or removing from it..

`StringSetTest.java` also includes class `StringSetTest` (reproduced below) with a `main` method to test your implementation.

```
class StringSetTest {

  public static void main(String[] args) {
    StringSet ws = new StringSet();
    // Test add()
    ws.add("pig"); ws.add("dog"); ws.add("cow");
    ws.add("cat");
    // Test size
    System.out.print("Set has "+ws.size()+" elements: ");
    // Test put()
    ws.put(); System.out.println();
    // Test contains() (uses add())
    ws.add("cow"); ws.add("ant");
    if (ws.contains("ant"))
      System.out.println("Set contains ant");
    else
      System.out.println("Set does not contain ant");
    if (ws.contains("goat"))
      System.out.println("Set contains goat");
    else
      System.out.println("Set does not contain goat");
    // Test remove() (uses size(), put())
    ws.remove("ant"); ws.remove("pig"); ws.remove("cow");
    ws.remove("pig");
    System.out.print("Set has "+ws.size()+" elements: ");
    ws.put(); System.out.println();
    ws.remove("cat"); ws.remove("cow"); ws.remove("dog");
    ws.remove("dog");
    System.out.print("Set has "+ws.size()+" elements: ");
    ws.put();
  }
}
```

If your implementation is correct you should see the following output:

```
  Set has 4 elements: cat cow dog pig
  Set contains ant
  Set does not contain goat
  Set has 2 elements: cat dog
  Set has 0 elements:
```

The indicated output is an indicator, not a guarantee, of correctness, Of course, the order in which set elements appear is not unique, and your program may well print them in a different order.

You may comment out parts of `main` while you are testing, but you must reinstate it fully at the end, leaving it exactly as provided.